

# Strategyproof Peer Selection

Haris Aziz<sup>a</sup>, Omer Lev<sup>b</sup>, Nicholas Mattei<sup>a</sup>, Jeffrey S. Rosenschein<sup>c</sup>, and Toby Walsh<sup>a</sup>

<sup>a</sup>Data61/NICTA and UNSW, Sydney, NSW 1466, Australia; <sup>b</sup>University of Toronto, Toronto, Ontario M5S 3G4, Canada; <sup>c</sup>The Hebrew University of Jerusalem, Jerusalem 91904, Israel

Peer review, evaluation, and selection is the foundation on which modern science is built. Funding bodies the world over employ experts to study and select the best proposals of those submitted for funding. The problem of peer selection, however, is much more universal: a professional society may want give a subset of its members awards based on the opinions of all the members; an instructor for a MOOC or online course may want to crowdsource grading; or a marketing company may select ideas from group brainstorming sessions based on peer evaluation. We make three fundamental contributions to the study of procedures or mechanisms for peer selection, a specific type of group decision making problem studied in computer science, economics, political science, and beyond. First, we detail a novel mechanism that is strategyproof, i.e., agents cannot benefit themselves by reporting insincere valuations, in addition to other desirable normative properties. Second, we demonstrate the effectiveness of our mechanism through a comprehensive simulation based comparison of our mechanism with a suite of mechanisms found in the computer science and economics literature. Finally, our mechanism employs a randomized rounding technique that is of independent interest, as it can be used as a randomized method to addresses the ubiquitous apportionment problem that arises in various settings where discrete resources such as parliamentary representation slots need to be divided fairly.

peer review | peer selection | social choice | allocation | apportionment

Since the beginning of civilization, societies have been selecting small groups from within. Athenian society, for example, selected a random subset of citizens to participate in the Boule, the council of citizens which ran daily affairs in Athens. Peer review, evaluation, and selection has been the foundational process by which scientific funding bodies have selected the best subset of proposals for funding [1] and peer evaluation is becoming increasingly popular and necessary to scale grading in MOOCs [2, 3]. In these settings, however, we do not wish to select an arbitrary subset of size  $k$ , but the “best  $k$ ”, and we need, therefore, a procedure in which the candidates are rated according to the opinions of the group. In peer selection problems we are not seeking an external, “independent” agent to make choices, but desire a crowdsourced approach, in which participants are those making the selection. Mechanisms for peer selection and their properties receive considerable attention within economics, political science, and computer science [1, 4–9].

Our motivation comes from the US National Science Foundation (NSF) recent “mechanism design pilot,” which attempted to spread the review load amongst all the submitters of proposals [1, 10]. The program uses “reviewers assigned from among the set of PIs whose proposals are being reviewed.” Reviewers’ own proposals get “supplemented with ‘bonus points’ depending upon the degree to which his or her ranking agrees with the consensus ranking ([11], Page 46).” This mechanism used by the NSF is not strategyproof; reviewers are incentivized to guess what others are thinking, not provide their

honest feedback (an idea supported by [12]). Removing the bonus may be worse, as reviewers would then be able to increase the chance of their own proposal being accepted by rating other proposals lower [13]. In either case, reviewers can benefit from reporting something other than their truthful values. When agents have the incentive to misrepresent their truthful reports, the effect on the results of the aggregation or selection mechanism can be problematic. Indeed, in a comprehensive evaluation of the peer review process, Wenneras and Wold [14] wrote in *Nature* that “...the development of peer-review systems with some built-in resistance to the weakness of human nature is therefore of high priority.”

We propose a novel strategyproof (or impartial) mechanism, where agents may never gain by being insincere. There are many reasons to prefer a mechanism which is strategyproof: first, the mechanism does not favor “sophisticated” agents who have the expertise to behave strategically. Second, agents with partial or no knowledge about the rankings of other agents are not disadvantaged when using a strategyproof mechanism. Third, normative properties of a mechanism typically assume a sincere behavior. If agents act strategically, we may lose some desirable normative properties. Fourth, it is easier to persuade people to use a strategyproof mechanism than one which can be (easily) manipulated. Note that while strategyproofness does not handle all biases agents may have, it eliminates an obvious “weakness in human nature.”

To achieve strategyproofness we could use a lottery (as in the Athenian democracy). However, this method does not select based on merit. A different option is to use a mechanism based on a voting rule. However, following Gibbard and Satterthwaite [15, 16], any “reasonable” mechanism based on voting will not be strategyproof unless it is a dictatorship. Another option is to use a mechanism like the Page Rank algorithm that use Markov chains to compute a ranking of the agents [17]. However, such mechanisms are also not strategyproof.

**Contributions.** First, we propose a novel peer selection mechanism (Exact Dollar Partition) that satisfies several desirable axiomatic properties including strategyproofness and two natural monotonicity properties. Second, we conduct a detailed experimental comparison with previously introduced strategyproof mechanisms with regard to their ability to recover the “ground truth”. Our experiments demonstrate that Exact Dollar Partition selects more high quality agents more often, selects more high quality agents in the worst case, and has more consistent quality than any other strategyproof mechanism in the literature. Third, our mechanism uses a novel randomized apportionment subroutine to fairly round selected fractional group sizes to integers. This subroutine is interesting in its own right as it provides a compelling solution to the

fundamental problem of apportionment: allocating representatives or resources in proportion to the group sizes or demands. Young [18] motivates the problem as follows: “*This surprisingly difficult problem has concerned statesmen, political analysts and mathematicians for over two hundred years.*”

**Discussion and Related Work.** Peer review and peer selection are the cornerstone of modern science and hence, the quality, veracity, and accuracy of peer review and peer evaluation is a topic of interest across a broad set of disciplines. Most empirical evaluations of peer review and peer selection study the effectiveness and limits of the system; typically by assembling large corpora of peer reviewed proposals and cross examining them with new panels or review processes [19, 20]. Questions of bias, nepotism, sexism, cronyism, among other issues, have received extensive coverage, and have been substantiated to varying degrees, in the literature [14, 19, 21]. However, a consistent conclusion in all of these works is that, in order to decrease the role of chance and/or any systematic bias, the community needs to broaden the base of reviewers. Indeed, one way for the results of the review process to reflect the views of the entire scientific constituency and provide more value to community is to increase the number of reviewers [22, 23]. The key scientific question lies in finding a mechanism that allows for crowdsourcing the work of reviewing, without compromising the incentives and quality of the peer review and selection process.

The criticism that prominent peer selection mechanisms such as those under consideration by American and European funding bodies [1, 10] are *not* strategyproof [13] has underscored the need to devise mechanisms with better incentive properties. The literature most directly relevant to this article is a series of papers on strategyproof (impartial) selection [4, 7]. We survey these mechanisms in the next section. Most of the work on strategyproof peer selection focuses on the setting in which agents simply approve (nominate) a subset of agents [4, 6, 7, 24] with the latter three restricting attention to the setting in which exactly one agent is selected ( $k = 1$ ). Kurakowa et al. [8] presented an interesting strategyproof mechanism (Credible Subset) that performs well when each agent reviews a very small number of agents relative to the total number of agents. Other recent work focused on tradeoffs between different axioms concerning peer selection [25, 26].

The peer selection problem is closely related to peer-based grading/marking [2, 17, 23, 27–30] especially when students are graded based on percentile scores. For peer grading, mechanisms have been proposed that make a student’s grade slightly dependent on the student’s grading accuracy (see e.g., [17] and [10]). However such mechanisms are not strategyproof as one may alter one’s reviews to obtain a better personal grade.

## Setup and Survey of Existing Mechanisms

We have a set  $N$  of agents  $\{1, \dots, n\}$ . An agent, depending on the setting, evaluates some  $0 \leq m \leq n - 1$  of the other agents. Based on these messages, around  $k$  agents are selected. Each agent reports a valuation (review) over the other agents (proposals). These messages could be cardinal valuations  $v_i(j)$  for agent  $i$ ’s valuations of agent  $j$ , or they could be a weak order (i.e., some agents may have the same rank) reported by agent  $i$  of agents in  $N \setminus \{i\}$ , which may be transformed to cardinal valuations using a scoring rule. Some mechanisms,

such as Credible Subset, may not always return a size of exactly  $k$  even if the *target* size is  $k$ .

A particular family of mechanisms which we will deal with are based on *partitioning*. The general idea of partitioning based mechanisms is to divide the agents into a set of clusters  $\mathcal{C} = \{C_1, \dots, C_\ell\}$ . This partition can be done using either a random or predetermined process, without adding randomness to the process. We will often abuse notation and refer to the value that one cluster has for another cluster  $v_{C_i}(C_j)$ ; the valuation of all agents in  $C_i$  for the agents in  $C_j$ :  $\sum_{r \in C_i, j \in C_j} v_r(j)$ . We will assume that cluster sizes are such that selection from them is not a problem: for all  $1 \leq i \leq \ell$ ,  $k \leq |C_i|$ .

**Mechanisms.** There are three prominent mechanisms for peer selection that appear in the literature. **Vanilla:** select the  $k$  agents with highest total value based on their reviews by other agents (as done today, for example, in many scientific conferences). Vanilla is not strategyproof; unselected agents have an incentive to lower the reported valuations of selected agents. **Partition:** divide the agents into  $\ell$  clusters and select a preset number of agents from each cluster, typically  $k/\ell$ , according to the valuations of the agents *not* in that cluster. This class of mechanisms is a straightforward generalization of the Partition mechanism [4] (and in an early version of [8]) and is strategyproof. **Credible Subset** [8]: let  $T$  be the set of agents who have the top  $k$  scores, as in Vanilla. Let  $P$  be the set of agents who do not have the top  $k$  scores but will make it to the top  $k$  if they do not contribute any score to other agents (hence  $|P| \leq m$ ). With probability  $(k + |P|)/(k + m)$ , Credible Subset selects a set of  $k$  agents uniformly at random from  $T \cup P$ , and with probability  $1 - (k + |P|)/(k + m)$ , it selects no one. The mechanism is strategyproof. There are a number of other mechanisms that are tailor-made for  $k = 1$  and when agents only mark approval of a subset of agents: Partition [7]; Permutation [6]; and Slicing [24].

In designing our mechanism, we were inspired by a prominent strategyproof mechanism for dividing a *continuous resource* from the economics literature. **Dividing a Dollar**[31]: Each agent  $i$  has a value  $v_i(j)$  of his estimation of how much of the resource  $j$  should receive, and these values are normalized so that  $\sum_{j \in N \setminus \{i\}} v_i(j) = 1/n$ . Then the *Dollar share* of each agent  $i$  is  $x_i = \sum_{j \in N \setminus \{i\}} v_j(i)$ .

**Properties of Mechanisms.** We consider some basic axioms of peer selection mechanisms: (i) *Non-imposition*: for any target set  $W$ , there is a valuation profile and a randomization seed that achieves  $W$ ; (ii) *Strategyproofness (Impartiality)*: agents cannot affect their own selection; (iii) *Monotonicity*: if an agent  $i$  is selected, then if in a modified setting in which that agent is reinforced (score or ordinal ranking of  $i$  is improved with respect to other agents) by some agents in  $N \setminus \{i\}$ , and no other agent’s value/ranking improved, then  $i$  will remain selected; (iv) *Committee Monotonicity*: if  $W$  is the outcome when the target set size is  $k$ , then the agents in  $W$  are still selected if the target set size is  $k + 1$ . These final two properties regard probability of selection when algorithms involve randomization.

## Exact Dollar Partition

The algorithm EXACT DOLLAR PARTITION is formally described in the algorithm box. Broadly, it works as follows:

**EXACTDOLLARPARTITION( $N, v_i$ )**

Input: Set of agents  $N$ .  
Set of valuations for each agent  $i, v_i$ .

Output: Set of winning agents  $W$ .

- 1: Initialize  $W \leftarrow \emptyset$
- 2: Generate a partition  $\{C_1, \dots, C_\ell\}$  of  $N$  where the difference between the sizes of any two clusters is at most 1.
- 3: Each  $i \in N$  reviews  $m$  agents outside  $C(i)$ , where  $C(i)$  is the cluster of agent  $i$ .
- 4: Ensure  $\sum_{j \notin C(i)} v_i(j) = 1$  by normalizing.
- 5:  $x_i$ , the value of a cluster  $C_i$ , is defined as:

$$x_i \leftarrow \frac{1}{n} \times \sum_{j \in C_i, j' \notin C_i} v_{j'}(j).$$

- 6: Using the  $x_i$  values, we compute the number of agents  $t_i$  to be chosen from each cluster  $C_i$ . First let each share  $s_i \leftarrow x_i \cdot k$  for each  $i \in \{1, \dots, \ell\}$ .
- 7:  $(t_1, \dots, t_\ell) \leftarrow \text{ALLOCATIONFROMSHARES}(s_1, \dots, s_\ell)$  where  $(t_1, \dots, t_\ell)$  are the number of agents to be allocated from each cluster.
- 8: For each  $i \in C(i)$ , the score of agent  $i$  is  $\sum_{i' \notin C(i)} v_{i'}(i)$ .
- 9: Select  $t_j$  agents with the highest scores from each cluster  $C_j$  and place them in set  $W$ .
- 10: **return**  $W$

agents are partitioned such that the sizes of clusters are equal or as near as possible, with difference at most 1. Each agent  $i \in N$  assigns a value  $v_i(j)$  to each agent  $j$  that is among the  $m$  agents that  $i$  reviews, none of which are in  $i$ 's cluster. Agent  $i$  may directly give a cardinal value to the agents they review or the cardinal value may be obtained by a scoring function that converts the ordinal ranking given by  $i$  to cardinal values. In either case, the values that  $i$  gives are normalized so that agent  $i$  assigns a total value of 1 to agents outside their own cluster. Based on the values from agents outside the cluster we assign the normalized weight (Dollar Share)  $x_j$  to each cluster  $C_j$ . Based on each Dollar Share  $x_i$ , each cluster has a possibly real valued quota  $s_i = x_i \cdot k$ . If all  $s_i$ 's are integers, then each  $s_i$  is the quota of cluster  $C_i$ , i.e., the top graded  $s_i$  agents are selected from cluster  $C_i$ . If not all  $s_i$  are integers, then we use the function **ALLOCATIONFROMSHARES** in line 7 (detailed in the next section) to enumerate discrete cluster allocations in which each cluster gets an allocation of either  $\lfloor s_i \rfloor$  or  $\lceil s_i \rceil$  is used. **ALLOCATIONFROMSHARES** then computes a probability distribution over these discrete allocations, requiring at most,  $\ell$  such allocations, so that the expected quota for each cluster will be exactly  $s_i$ . We draw a discrete allocation  $(t_1, \dots, t_\ell)$  using the distribution computed in **ALLOCATIONFROMSHARES** and select exactly the  $t_i$  agents from each cluster  $C_i$  with highest score.

We defer our proofs and analysis of the properties of the function **ALLOCATIONFROMSHARES** to the next section. For the analysis of the overall mechanism, it is enough to assume it chooses an allocation with exactly  $k$  shares from a probability space constructed so that the expected share of each cluster  $j$  is  $s_j$ . As no agent is treated differently in the mechanism, Exact Dollar Partition is anonymous and satisfies non-imposition. In the supplemental material, we show that Exact Dollar

Partition retains key normative properties that are important for any peer selection setting.

**Theorem 1** *Exact Dollar Partition is strategyproof.*

**Remark 2** *Exact Dollar Partition is not just strategyproof but even group-strategyproof if manipulating coalitions involve agents from the same cluster (so potentially colluding agents, e.g., with conflict of interest, can be put in the same cluster).*

**Theorem 3** *Exact Dollar Partition is monotonic.*

**Theorem 4** *Exact Dollar Partition is committee monotonic.*

## A Randomized Apportionment Rule

The randomized allocation technique used for Exact Dollar Partition is of independent interest since it addresses the classic apportionment problem in a randomized way. Consider the problem in which  $\ell$  disjoint groups are to be allocated a given number of slots  $k < n$  in proportion to the group sizes. The problem is ubiquitous in apportionment settings such as proportional representation of seats in the US congress, European Parliament, and the German Bundestag as well as various other committee selection settings [18, 32–36]. It has been studied in political science, economics, operations research and computer science.

In these settings, each group  $i$  has a quota  $s_i$  with  $\sum_{i=1}^n s_i = k$ , which we call its *target quota*. Since  $s_i$  may not be an integer, we have to resort to *apportionment* which means that in order to allocate exactly  $k$  slots, some group may be assigned an integer quota slightly more or less than its target quota.

Numerous apportionment procedures have been introduced in the literature including the methods of Hamilton, Jefferson, Webster, Adams, and Hill [32]; each with its own drawbacks. In fact, *no* deterministic apportionment procedure can satisfy a group of three minimal axioms [32]: (1) *Quota Rule*, each group should get quota that is the result of the target quota being rounded up or down; (2) *Committee Monotonicity*, if  $k$  increases then then the quotas do not decrease; and (3) *Monotonicity*, if group  $i$ 's size increases and group  $j$ 's size decreases by the same margin, then then there is no transfer of quota from  $i$  to  $j$ . We call discrete quota allocations that satisfy the quota rule and allocate exactly  $k$  slots *nice allocations*.

**Curse of Determinism: The Need for Randomization.** In view of the result by Balinski and Young [32], we first demonstrate that randomization is a necessary feature of an apportionment mechanism in order to maintain both exactness and strategyproofness. Therefore, any deterministic and strategyproof method of using fractional quotas to derive integer quotas can result in outcomes that are not of the target size (e.g., [37]).

**Theorem 5** *No deterministic rounding of quotas that guarantees selection of  $k$  agents is strategyproof.*

Hence, we resort to randomization to achieve *ex ante* fairness and the target size. Using randomization, our goal is to ensure that the target number of total agents chosen is  $k$  in expectation and exactly  $k$  *ex post*. We will additionally require that the integer quota allocation returned by the lottery is a nice allocation. Randomization has been used in various settings such as voting and fair allocation of indivisible goods to achieve *ex ante* as well as *procedural* fairness [38–40].



One possible way to achieve the appropriate randomization is to enumerate all the feasible discrete quota allocations and then solve equations to find the probability distribution over these quota allocations. If such a probability distribution exists, the method outlined involves enumerating an exponential number of such quota allocations that is computationally infeasible if the number of groups is large (for example, in US elections,  $\ell$  is 50). Hence, some suggested approaches to this problem require multiple rounds of randomization and also do not enumerate the possible ex post outcomes (there may be an exponential number of them) [41]. Previously, a stochastic apportionment rule was presented that achieves the quota requirements [42] by two randomizations, one of them using a stochastic continuous variable. This means that not only does it not involve a probability distribution over discrete nice allocations, hence it cannot be used to achieve fairness via repeated representation, but due to computers not being able to reproduce truly continuous values, this may compromise strategyproofness.

In view of these challenges, we present a simple method **ALLOCATIONFROMSHARES** that achieves the target quotas, relies on probability distribution over *linear number* of nice allocations, and the probability distribution can be computed in linear time, and requires minimal randomization (only one round). Our randomized procedure can be easily de-randomized in repetitive settings. In frequently repeated allocation settings, one could use the nice allocations computed by **ALLOCATIONFROMSHARES** (at most  $\ell$ , compared to the potential exponential number) in a way such that the allocations have the same frequency as the probability distribution computed by the algorithm. In this sense, our randomized apportionment routine has an advantage over other proposed methods.

Informally, our method proceeds gradually from quotas which need to be rounded up with low probability, while keeping an eye on our two main constraints: not rounding up a quota too much, on one hand, while not being left with not enough probability for allocations with quotas which need to be rounded up with high probability.

**Example 6** We give an example of the working of **ALLOCATIONFROMSHARES** when a set of quotas are not integers. Suppose we have the following Dollar shares for  $\ell = 5$ , (1.1, 1.1, 1.3, 1.7, 1.8) and we wish to select  $k = 7$  agents. The  $\alpha$ , computed on line 5, is 2, and we start with  $low = 1; high = 5$ .

We begin considering the allocation (2, 2, 1, 1, 1). Since  $0.1 = s_1 - \lfloor s_1 \rfloor < \lceil s_5 \rceil - s_5 = 0.2$ , it will get a probability of 0.1. Now  $low = 2$ ,  $\bar{p} = 0.1$ , and we “slide” our allocation by one to the right, and look at (1, 2, 2, 1, 1). Since the second cluster has been rounded up with a probability of 0.1 in the previous allocation,  $s_2 - \lfloor s_2 \rfloor - p(v_2) = 0$ . Therefore this allocation is given probability 0,  $\bar{p}$  does not change and now  $low = 3$ . Moving to allocation (1, 1, 2, 2, 1), we see  $0.3 = s_3 - \lfloor s_3 \rfloor - p(v_3) > \lceil s_5 \rceil - s_5 - \bar{p} + p(v_5) = 0.1$ , so this allocation is given probability 0.1,  $\bar{p} = 0.2$ ,  $\alpha = 1$  and  $high = 4$ . We now look at (1, 1, 2, 1, 2), since  $0.2 = s_3 - \lfloor s_3 \rfloor - p(v_3) = \lceil s_4 \rceil - s_4 - \bar{p} + p(v_4) = 0.2$ , we give it the probability 0.2,  $\bar{p} = 0.4$ , and  $low = 4$ . Finally, we look at (1, 1, 1, 2, 2), giving it the probability  $s_4 - \lfloor s_4 \rfloor - p(v_4) = 0.6$ .

Overall, we have probability distribution [(2, 2, 1, 1, 1) : 0.1, (1, 1, 2, 2, 1) : 0.1, (1, 1, 2, 1, 2) : 0.2, (1, 1, 1, 2, 2) : 0.6].

**Theorem 7** *AllocationFromShares defines a distribution and the expected allocation of each cluster  $i$  is its share  $s_i$ .*

**ALLOCATIONFROMSHARES**( $s_1, \dots, s_\ell$ )

Input: A real allocation ( $s_1, \dots, s_\ell$ ) over  $\ell$  objects.

Output: A discrete allocation ( $t_1, \dots, t_\ell$ ) over  $\ell$  objects.

```

1: Sort and renumber ( $s_1, \dots, s_\ell$ ) according to size of
    $s_i - \lfloor s_i \rfloor$ , with  $s_1 - \lfloor s_1 \rfloor$  being minimal.
2: Let ( $p_1, \dots, p_\ell$ )  $\leftarrow$  (0, ..., 0) where  $p_i$  is the proba-
   bility of rounding up cluster  $i$ .
3: Let  $\bar{p} \leftarrow 0$ , the total probability allocated so far.
4: Let  $D \leftarrow \emptyset$ , where  $D$  maps: allocation  $\rightarrow$  probability.
5:  $\alpha \leftarrow \sum_{i=1}^{\ell} (s_i - \lfloor s_i \rfloor)$ 
6:  $low \leftarrow 1; high \leftarrow \ell$ 
7: while  $low \leq high$  do
8:   Let  $allocation \leftarrow (\lfloor s_1 \rfloor, \dots, \lfloor s_{low-1} \rfloor, \lceil s_{low} \rceil, \dots,$ 
      $\lceil s_{low+\alpha-1} \rceil, \lfloor s_{low+\alpha} \rfloor, \dots, \lfloor s_{high} \rfloor, \lceil s_{high+1} \rceil, \dots, \lceil s_\ell \rceil)$ 
     Where if  $low=1$ , we start with  $\lceil s_1 \rceil$ ; if  $high = \ell$ , we
     end with  $\lfloor s_{high} \rfloor$ ; and if  $\alpha = 0$ , we have only  $\lfloor s_{low} \rfloor$ .
9:    $prob \leftarrow 0$ 
10:   $prevLow \leftarrow low; prevHigh \leftarrow high$ 
11:  if  $\alpha = 0$  then
12:     $prob \leftarrow 1 - \bar{p}; high \leftarrow high - 1$ 
13:  else
14:    if  $s_{low} - \lfloor s_{low} \rfloor - p_{low} < \lceil s_{high} \rceil - s_{high} - \bar{p} + p_{high}$ 
      then
15:       $prob \leftarrow s_{low} - \lfloor s_{low} \rfloor - p_{low}; low \leftarrow low + 1$ 
16:    else
17:       $prob \leftarrow \lceil s_{high} \rceil - s_{high} - \bar{p} + p_{high}$ 
18:       $high \leftarrow high - 1; \alpha \leftarrow \alpha - 1$ 
19:    for all  $i$  such that  $prevLow \leq i < prevLow + \alpha$  or
       $prevHigh < i \leq \ell$  do
20:       $p_i \leftarrow p_i + prob$ 
21:       $\bar{p} \leftarrow \bar{p} + prob$ 
22:       $D \leftarrow D \cup (allocation \rightarrow prob)$ 
23: Select an allocation ( $t_1, \dots, t_\ell$ ) according to  $D$ .
24: return ( $t_1, \dots, t_\ell$ )

```

## Analytical Comparisons with Other Mechanisms

Though Exact Dollar Partition draws inspiration from Dividing a Dollar and Partition, there are key differences.

**Comparison with other Dollar Based Mechanisms.** Although Exact Dollar Partition is partly based on the Dollar mechanism for dividing a bonus (division of a divisible item between agents), it is more desirable than other mechanisms one can construct based on the Dollar framework. Consider the following possible adaptations of the Dollar framework and their shortcomings. **Dollar Raffle** computes the relative fractions of how much of a dollar each agent should get via the Dollar mechanism [31]. Using these shares as probabilities, repeatedly select randomly an agent according to its dollar share probabilities until  $k$  different agents are selected. **Dollar Partition Raffle** takes the Dollar shares of the clusters in Dollar Raffle and uses these shares to define a probability distribution over the clusters. A cluster is drawn with respect to the cluster Dollar probabilities and the next best agent, based on reviews of agents outside the cluster, is selected, until  $k$  different agents are selected. **Top Dollar** selects agents with maximum Dollar shares.<sup>1</sup>

Both Dollar Raffle and Dollar Partition Raffle have a non-

<sup>1</sup> Vanilla is equivalent to Top Dollar when agents' valuations are normalized.

zero probability of selecting the  $k$  worse agents. While Top Dollar is not strategyproof for any  $k$ , Dollar Raffle and Dollar Partition Raffle are strategyproof for  $k = 1$ . None, however, are strategyproof for  $k > 1$ . We can also show the following theorem (which can be extended to the various mechanisms mentioned for  $k = 1$  (e.g., [6])).

**Theorem 8** *Dollar Raffle, Dollar Partition Raffle and Top Dollar are not strategyproof for  $k > 1$ .*

**Comparison with Partition Mechanisms.** Exact Dollar Partition seems similar to the Partition mechanism but while Partition must preset the number of agents to be selected from each cluster, Exact Dollar Partition *relies on the peer reviews* to decide the number of agents to be selected from each cluster. This difference allows Exact Dollar Partition to have more consistent performance, no matter the clustering. Hence, in contrast to Exact Dollar Partition, the rigidity of Partition means that it may not choose a large proportion of the best agents even if agents have unanimous valuations.

**Example 9** *Consider the setting in which  $N = \{1, \dots, 18\}$ ,  $k = 6$ , and  $\ell = 3$ . Let the clusters be  $C_1 = \{1, \dots, 6\}$ ,  $C_2 = \{7, \dots, 12\}$ ,  $C_3 = \{13, \dots, 18\}$ .  $C_1$  puts all its weight on  $C_2$ , equally dividing its points between 7, 8, ..., 12, with a slight edge to 7 and 8,  $C_2$  and  $C_3$  put all the weight on  $C_1$ , dividing their points between 1, 2, 3 and 4. Now Partition will choose 1, 2, 7, 8, 13, 14 where everyone thinks that 1, 2, 3, 4, 7, 8 are the best. Exact Dollar Partition will select exactly that set. Moreover, if we increase the number of clusters, the disparity between Exact Dollar Partition and Partition only grows.*

Partition, in contrast to Exact Dollar Partition, performs poorly ex post<sup>2</sup> if the clusters are lopsided, with some cluster containing all good agents and other clusters containing low value agents. One natural fix is to deliberately choose a balanced partition where the weight of a cluster is based on the ratings of agents outside the cluster and we choose a clustering that minimize the difference between the cluster weights. However for this and various notions of balanced partitions, computing the most balanced partition is NP-hard. What is even more problematic is that if we choose a balanced partition, the resulting mechanism is not strategyproof.

We point out that there are instances where Partition may perform better than Exact Dollar Partition even if the rankings of the agents are unanimous. Consider a case where a highly preferred agent is in the same group as the lowest preferred agents, while other groups only contain medium preferred agents. In that case the weight of the cluster with the highest preferred agent might be so high that lowest ranked agents might also be selected. The normalization of scores entailed in Exact Dollar Partition causes a certain loss of information and granularity compared to the other mechanisms. However, even in the example above, Exact Dollar Partition will ensure that when agents have highly correlated or unanimous preferences, the agent(s) that are unanimously on the top will be selected, even if some low-ranked agents are also selected.

## Experimental Comparison with Other Mechanisms

Using Python and extending code from PREFLIB [43] we have implemented the Exact Dollar Partition, Credible Subset,

Partition, Dollar Raffle, Dollar Partition Raffle, and Vanilla peer selection mechanisms. All the code developed for this project is available open sourced at . In all simulations there are many parameters to consider that can drastically affect the outcome (see e.g., [44]). By focusing on a target domain, the NSF Mechanism Design Pilot [10, 11], we can draw focused conclusions from our simulations. In 2014, this program had  $n = 131$  proposals, with each submitter reviewing  $m = 7$  other proposals, broken into  $\ell = 4$  clusters. The acceptance numbers are not broken out from the global  $\approx 20\%$  acceptance rate, so we use this as the acceptance rate.

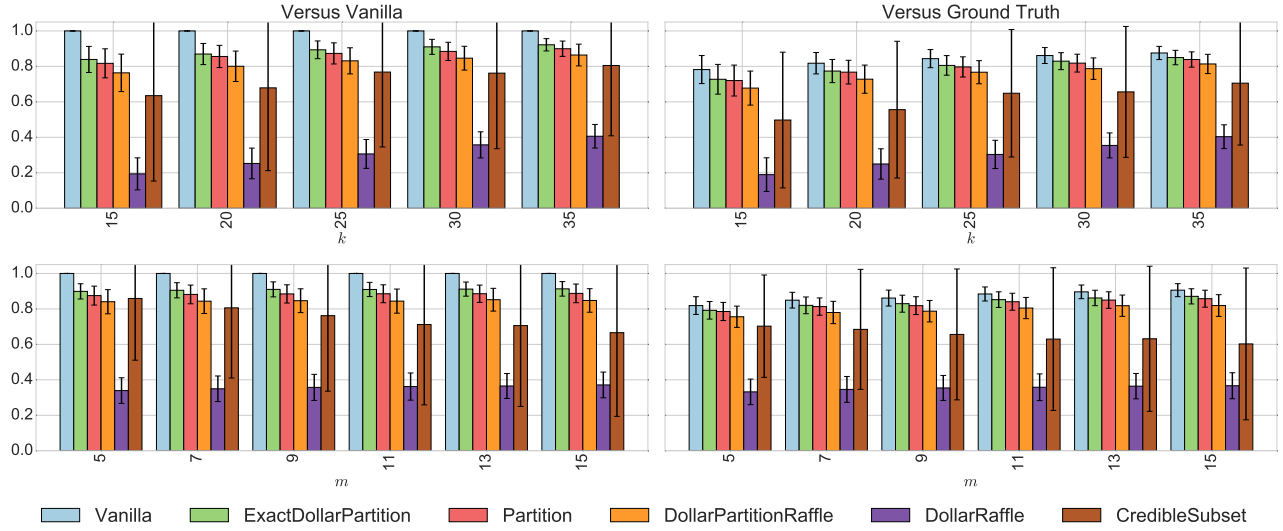
**Experimental Setup.** To create NSF-like data we generate a sparse  $N = \{1, \dots, n\}$  scoring matrix (profile) using a *Mallows Model* to generate the ordinal evaluation [45, 46]. Mallows models are parameterized by a *reference order* ( $\sigma$ ) and a *dispersion parameter* ( $\phi$ ). The reference order  $\sigma$  can be thought of as the underlying ground truth; the NSF mechanism assumes implicitly that there is some true ordering of proposals that the reviewers are attempting to guess. Intuitively, the dispersion parameter  $\phi$  is the probability of committing ranking error by swapping neighboring elements according to  $\sigma$ , where  $\phi = 0$  means that no agent ever commits an error and  $\phi = 1.0$  means that orderings are generated uniformly at random [47]. Mallows models are used when each agent is assumed to have the same reference ranking subject to a common, independent noise model. Each agent  $i \in N$  ends up with a ranking  $rank(i, j) \rightarrow \{0, \dots, m-1\}$  over each agent  $j \in N$  where  $rank(i, j) = 0$  means  $j$  is the highest reviewed proposal by  $i$ .

Each agent reviews  $m$  of the  $n$  proposals and is also reviewed by  $m$  other agents. Agents are clustered under the constraint that each agent reviews  $m$  agents *outside* his cluster. We call a reviewer assignments satisfying these constraints a balanced  $m$ -regular assignments. To maximize inter-cluster comparison, we want the  $m$  reviews provided by agent  $i$  to be balanced among the clusters (less  $C_i$ ) so agent  $i$  in cluster  $C_i$  reviews in total  $\frac{|C_i| \cdot m}{\ell - 1}$  agents from each other cluster. We generate this assignment randomly and as close to balanced as possible.

We generate a sparse  $n \times n$  score matrix by: drawing a balanced  $m$ -regular assignment; generating a complete ordinal ranking using a Mallows model for agent  $i$ ; removing all candidates from  $i$ 's ordinal ranking not assigned to  $i$ ; and assigning score  $m - rank(i, j)$  to each agent  $j$  that  $i$  ranks (known as the Borda score). This process mimics the underlying assumption of the NSF mechanism in that, if a reviewer were to see all proposals, they could strictly order the complete set. This process leaves us with a sparse  $n \times n$  score matrix which obeys an  $m$ -regular assignment of agents partitioned into  $\ell$  clusters.

We use two different orderings to evaluate the performance of all the mechanisms presented: the ground truth (GT) ordering and the ordering selected by vanilla (V). Since a Vanilla like mechanism is used in many settings, it is fitting to see how well the strategyproof mechanisms approximate it (though we assume it is not manipulated by the participants despite not being strategyproof). This allows us to understand the “price” we are paying for strategyproofness. Formally, let  $W, W'$  be the winning sets returned by two mechanisms, we measure the similarity of  $W$  to  $W'$  by  $|W \cap W'|/k$ . For  $n = 130$  and  $s = 1000$ , we looked at a “NSF Like” space with  $k \in \{15, 20, 25, 30, 35\}$ ,  $m \in \{5, 7, 9, 11, 13, 15\}$ ,  $\phi \in \{0.0, 0.10, 0.20, 0.35, 0.50\}$ , and  $\ell \in \{3, 4, 5, 6\}$ .

<sup>2</sup>For high stakes outcomes, we want a mechanism that performs well on average and never returns an especially bad outcome.



**Fig. 1.** Performance of the six mechanisms surveyed in this paper as we vary settings to  $k$  (top row) and  $m$  (bottom row) as measured in selected percentage of the Vanilla set (V, left) and ground truth ordering (GT, right). Note that the colors of the bars are in the same order as the table key. For both figures we have set  $n = 130$ ,  $\ell = 4$ , and  $\phi = 0.5$ ; for the top row, we set  $m = 9$  as we varied  $k$  and for the bottom row we set  $k = 30$  as we vary  $m$ . Across the tested range ExactDollarPartition outperforms all other mechanisms when measured against the V or GT ordering with a lower standard deviation. As we increase the value of  $m$  or  $k$ , ExactDollarPartition improves its performance at a faster rate and more consistently than any other mechanism.

**General Results.** It is hard to directly compare results for Credible Subset due to the large probability of returning an empty set under the given parameters. In fact, counter to intuition, Credible Subset performs worse as we increase the number of reviews because this *increases* the chance of returning an empty set (see, e.g., Figure 1.) This problem is not easily overcome; removing the ability to return an empty set means Credible Subset is no longer strategyproof. When Credible Subset does return a set, it performs very well, on par with Vanilla. However, the minimum (0 in some cases), average, and standard deviation for Credible Subset are all unacceptable for practical implementation.

Throughout the testing Exact Dollar Partition strictly outperforms, for all parameter settings, the other Dollar based mechanisms. Hence, we conclude that the extra steps developed for Exact Dollar Partition are necessary and result in dramatically increased performance. In this discussion we will focus on comparing the performance of Exact Dollar Partition and Partition only, as these two mechanisms are the top two performing mechanisms which are also strategyproof. Broadly, we find that Exact Dollar Partition outperforms Partition. On average, Exact Dollar Partition selects between 0.5% and 5% more top agents, measured against V or GT. It does this with between 3% and 25% lower standard deviation, and selects as many or, in the most extreme cases, up to five more top performing agents. This improvement in the worst case means Exact Dollar Partition is an improvement of up to 25% when the clustering of the agents is lopsided. Hence, Exact Dollar Partition is the best in our experiments: it selects more top agents, more often, with better worst case performance and lower variance than any other strategyproof mechanism.

**Varying Noise( $\phi$ ) and Clusters ( $\ell$ ).** Varying the noise parameter  $\phi$  has little to no effect on the approximation to V for all mechanisms. This is not surprising as all mechanisms receive the same (noisy) information. When approximating GT, the value of  $\phi$  has negligible effect on the performance of the mechanisms

unless  $\phi \geq 0.95$ ; for the remainder of the discussion we fix  $\phi = 0.5$ . Varying the setting of  $\ell$  we see that all mechanisms perform best with respect to GT when we set  $\ell = 5$  and with respect to V when  $\ell = 3$ ; with a decrease in performance as we continue to increase  $\ell$ . No matter the setting, increasing the number of clusters hurt Vanilla and Exact Dollar Partition's performance the least, i.e., their performance decreases less quickly than the other mechanisms. For the remainder we set  $\ell = 4$  as was done for the NSF pilot.

**Varying the Number of Selections ( $k$ ) and Reviews ( $m$ ).** Figure 1 captures our metrics as we vary the number of selections  $k$ , in the top row, and the number of reviews per item  $m$ , in the bottom row. Varying the setting to  $k$  we observe fairly consistent performance by the mechanisms with Exact Dollar Partition maintaining a 1.5% to 3% advantage. The biggest percentage wise advantages are found when  $k = 15$ , where Exact Dollar Partition selects up to two better agents more according to V, in the worst case, resulting in a  $\approx 25\%$  improvement. In the worst case, up to two more top agents according to GT are selected by Exact Dollar Partition than Partition. Because both mechanisms perform worse (in absolute terms) than they do as measured by V, this translates to a 10–20% increase in performance for Exact Dollar Partition when  $k$  is small and a 5–10% increase when  $k$  is large. Measured against both V and GT draw the general conclusion that, as we increase  $k$ , ExactDollarPartition increases its advantage over Partition.

For the most NSF like setting,  $n = 130$ ,  $k = 30$ ,  $\ell = 4$ ,  $\phi = 0.5$  and sweeping  $m \in \{5, 7, 9, 11, 13, 15\}$  is depicted as the bottom row of Figure 1. Looking closely at the numbers as measured against V we see that Exact Dollar Partition performs 2.6 to 3.0% better on average, i.e., one better agent. ExactDollarPartition does this with a nearly 20% smaller standard deviation, always selecting at least one more and up to three more top agents (15%) in the worst case. This pattern is similar across settings to the other parameters as vary  $m$ ; ExactDollarPartition performs consistently better. as we in-

crease the number of reviews, ExactDollarPartition continues to outpace Partition. Compared to the performance of Vanilla, ExactDollarPartition selects about one worse agent on average, up to two worse agents in the worst case ( $\approx 7\%$ ). Hence, the loss in performance we see for moving to a strategyproof mechanism is similar to the worse performance of Partition compared to ExactDollarPartition.

**ACKNOWLEDGMENTS.** The authors wish to thank Allan Borodin, Manuel Cebrian, Serge Gaspers, Ian Kash, Julian Mestre, and Hervé Moulin for useful comments. Data61 (formerly known as NICTA) is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. This research has also been partly funded by Microsoft Research through its PhD Scholarship Program, Israel Science Foundation grant #1227/12, and NSERC grant 482671. This work has also been partly supported by COST Action IC1205 on Computational Social Choice.

1. Hazeliigg GA (2013) Dear Colleague Letter: Information to Principal Investigators (PIs) Planning to Submit Proposals to the Sensors and Sensing Systems (SSS) Program October 1, 2013, Deadline (NSF Website, <http://www.nsf.gov/pubs/2013/nsf13096/nsf13096.jsp>).
2. Piech C et al. (2013) Tuned models of peer assessment in MOOCs in *Proceedings of the 6th International Conference on Educational Data Mining (EDM)*. (Memphis, Tennessee), pp. 153–160.
3. Luo H, Robinson AC, Park JY (2014) Peer grading in a MOOC: Reliability, validity, and perceived effects. *Journal of Asynchronous Learning Networks* 18(2).
4. Alon N, Fischer F, Procaccia AD, Tennenholtz M (2011) Sum of us: Strategyproof selection from the selectors in *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*. pp. 101–110.
5. Lakhani KR, Garvin DA, Lonstein E (2010) Topcoder (a): Developing software through crowdsourcing. *Harvard Business Review*.
6. Fischer F, Klimm M (2014) Optimal impartial selection in *Proceedings of the 15th ACM Conference on Economics and Computation (ACM-EC)*. (ACM Press), pp. 803–820.
7. Holzman R, Moulin H (2013) Impartial nominations for a prize. *Econometrica* 81(1):173–196.
8. Kurokawa D, Lev O, Morgenstern J, Procaccia AD (2015) Impartial peer review in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*. (AAAI Press).
9. Roos M, Rothe J, Scheuermann B (2011) How to calibrate the scores of biased reviewers by quadratic programming in *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*. pp. 255–260.
10. Merrifield MR, Saari DG (2009) Telescope time without tears: a distributed approach to peer review. *Astronomy & Geophysics* 50(4):4–16.
11. Foundation TNS (2014) Report to the national science board on the national science foundation's merit review process fiscal year 2014, (The National Science Foundation (USA)), Technical report.
12. Court D, Gillen B, McKenzie J, Plott CR (2015) Two information aggregation mechanisms for predicting the opening weekend box office revenues of films: Boxoffice Prophecy and guess of guesses, (California Institute of Technology), Technical Report SSWP-1412.
13. Naghizadeh P, Liu M (2013) Incentives, quality, and risks: A look into the NSF proposal review pilot. *arXiv preprint arXiv:1307.6528* pp. 1–10.
14. Wenneras C, Wold A (1997) Nepotism and sexism in peer-review. *Nature* 387:341–343.
15. Gibbard A (1973) Manipulation of voting schemes. *Econometrica* 41(4):587–602.
16. Satterthwaite MA (1975) Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory* 10(2):187–217.
17. Walsh T (2014) The PeerRank method for peer assessment in *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*. pp. 909–914.
18. Young HP (1994) *Equity: in Theory and Practice*. (Princeton University Press).
19. Cole S, Cole J, Simon G (1981) Chance and consensus in peer review. *Science* 214(4523):881–886.
20. Li D, Agha L (2015) Research funding, big names or big ideas: do peer-review panels select the best science proposals? *Science (New York, N.Y.)* 348(6233):434–438.
21. McNutt RA, Evans AT, Fletcher RH, Fletcher SW (1990) The effects of blinding on the quality of peer review: A randomized trial. *The Journal of the American Medical Association* 263(10):1371–1376.
22. Price E (2014) The NIPS experiment (<http://blog.mrtz.org/2014/12/15/the-nips-experiment.html>).
23. Joachims T, Raman K (2015) Bayesian ordinal aggregation of peer assessments: A case study on KDD 2015, (Cornell University), Technical report.
24. Bousquet N, Norin S, Vetta A (2014) A near-optimal mechanism for impartial selection in *Proceedings of the 10th International Workshop on Internet and Network Economics (WINE)*, Lecture Notes in Computer Science (LNCS).
25. Mackenzie A (2015) Symmetry and impartial lotteries. *Games and Economic Behavior* 94(1):15–28.
26. Berga D, Gjorgiev R (2014) Impartial social rankings. Working paper.
27. Alfaro LD, Shavlovsky M (2014) CrowdGrader: Crowdsourcing the Evaluation of Homework Assignments in *Proceedings of the ACM Technical Symposium on Computer Science Education (ACM-SIGCSE)*. pp. 415–420.
28. Kulkarni C, Wei K, Le H, Chia KD (2013) Peer and self assessment in massive online classes. *ACM Transactions on Computer Human Interaction (TOCHI)* 20(6):1–31.
29. Robinson R (2001) Calibrated peer review an application to increase student reading and writing skills. *The American Biology Teacher* 63(7):474–476.
30. Wright J, Thornton C, Leyton-Brown K (2015) Mechanical TA: Partially automated high-stakes peer grading in *Proceedings of the ACM Technical Symposium on Computer Science Education (ACM-SIGCSE)*.
31. de Clippel G, Moulin H, Tideman N (2008) Impartial division of a dollar. *Journal of Economic Theory* 139:176–191.
32. Balinski M, Young HP (1982) *Fair Representation*. (Yale University Press).
33. Balinski ML, Young HP (1980) The webster method of apportionment. *Proceedings of the National Academy of Sciences (PNAS)* 77(1):1–4.
34. Birkhoff G (1976) House monotone apportionment schemes. *Proceedings of the National Academy of Sciences (PNAS)* 73(3):684–686.
35. Mayberry JP (1978) Quota methods for congressional apportionment are still non-unique. *Proceedings of the National Academy of Sciences (PNAS)* 75(8):3537–3539.
36. Pukelshei F (2014) *Proportional Representation: Apportionment Methods and Their Applications*. (Springer).
37. Aziz H, Lev O, Mattei N, Rosenchein JS, Walsh T (2016) Strategyproof peer selection: Mechanisms, analyses, and experiments in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*.
38. Gibbard A (1977) Manipulation of schemes that mix voting with chance. *Econometrica* 45(3):665–681.
39. Budish E, Che YK, Kojima F, Milgrom P (2013) Designing random allocation mechanisms: Theory and applications. *American Economic Review* 103(2):585–623.
40. Bogomolnaia A, Moulin H (2001) A new solution to the random assignment problem. *Journal of Economic Theory* 100(2):295–328.
41. Gandhi R, Khuller S, Parthasarathy S, Srinivasan A (2006) Dependent rounding and its applications to approximation algorithms. *Journal of the ACM* 53(3):324–360.
42. Grimmet G (2004) Stochastic apportionment. *The American Mathematical Monthly* 111(4):299–307.
43. Mattei N, Walsh T (2013) Preflib: A library for preferences. [HTTP://WWW.PREFLIB.ORG](http://www.preflib.org) in *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT)*. pp. 259–270.
44. Popova A, Regenwetter M, Mattei N (2013) A behavioral perspective on social choice. *Annals of Mathematics and Artificial Intelligence* 68(1–3):135–160.
45. Mallows C (1957) Non-null ranking models. *Biometrika* 44(1):114–130.
46. Marden JI (1996) *Analyzing and Modeling Rank Data*, Monographs on Statistics and Applied Probability. (CRC Press) No. 64.
47. Lu T, Boutilier C (2011) Learning Mallows models with pairwise preferences in *Proceedings of the 28th International Conference on Machine Learning (ICML)*. pp. 145–152.



## Deferred Proofs

### A. Proof of Theorem 1

**Proof:** Suppose agent  $i$  is in cluster  $C_j$  of the generated partition. Agent  $i$  will be selected in  $W$  if and only if its rank according to the  $v$  scores given by agents outside of  $C_j$  is at least  $t_j$ . Therefore agent  $i$  can either manipulate by increasing  $t_j$  or by increasing its score relative to other agents in  $C_j$  given by agents outside  $C_j$ . Since agent  $i$  cannot affect the latter, the only way it can manipulate is by increasing  $t_j$ . We argue that agent  $i$  cannot change its *expected*  $t_j$  by changing its valuation  $v_i$  for agents outside the cluster. Note that  $i$  contributes a probability weight of  $1/n$  to agents outside  $C_j$  and zero probability weight to agents in  $C_j$ . Hence it cannot affect the value  $x_j$  of cluster  $C_j$ . As  $s_j$  is derived from  $x_j$ , agent  $i$  cannot affect  $s_j$ . Note that any agent in cluster  $C_j$  changing its report, not only is the expected  $t_j$  (which is the same as  $s_j$ ) does not change but the respective probabilities of getting  $\lfloor s_j \rfloor$  and  $\lceil s_h \rceil$  allocated do not change as well. The reason is that  $s_j$  is obtained as a unique convex combination of  $\lfloor s_i \rfloor$  and  $\lceil s_i \rceil$ .  $\square$

### B. Proof of Theorem 3

**Proof:** Let us compare the valuation profile  $v$  when  $i$  is not reinforced and  $v'$  when  $i$  is reinforced. The relative ranking of  $i$  is at least as high when  $i$  is reinforced. Since any decrease in valuation that an agent  $j$  in  $C(i)$  receives translates into the same increase in the valuation received by agent  $i$ , hence the total valuation that  $C(i)$  receives does not decrease and hence the number of agent to be selected from  $C(i)$  is at least as high as before.  $\square$

### C. Proof of Theorem 4

**Proof:** The only difference between running the algorithm for different target  $k$  values is when calculating the quota vector  $\vec{s}$ . However, if agent  $i$  in cluster  $C_j$  was selected, that means its ranking in the cluster  $C_j$  was above  $t_j$ . When  $k$  increases,  $s_j$  will only increase (as  $x_j$  remains the same), and hence so will  $t_j$ , ensuring that  $i$  will be selected again.  $\square$

### D. Proof of Theorem 5

We first prove the following related theorem.

**Theorem 10** *There is no anonymous deterministic allocation of quotas that guarantees selection of  $k$  agents.*

**Proof:** Let  $\ell = 3$ , with clusters being of equal size. All agents in cluster 1 rank agents in cluster 2 before any in cluster 3; agents in cluster 2 rank those in cluster 3 ahead of cluster 1; and those in cluster 3 rank agents in cluster 1 ahead of those in 2. So share of each cluster is equal, and for  $k < \ell$  there is no deterministic anonymous way to allocate quotas.  $\square$

**Lemma 11** *In a deterministic strategyproof allocation mechanism to select  $k$  agents from  $\ell$  clusters, the number of slots allocated to a cluster  $i$  with a share  $x_i$  will not change, regardless of the rest of the shares.*

**Proof:** Let  $x = (x_1, \dots, x_\ell)$  be a share allocation, under which cluster  $i$  receives  $y$  slots. Now, let  $x' = (x'_1, \dots, x'_{i-1}, x_i, x'_{i+1}, \dots, x'_\ell)$  be a different share allocation. We wish to show that the allocation to cluster  $i$  remains  $y$ .

We know  $\sum_{j \neq i} x_j = \sum_{j \neq i} x'_j$ . If  $\sum_{j \neq i} |k(x_j - x'_j)| \leq 2$ , this means a single agent can cause the change. As the share values do not contain data on actual agents votes, that single agent could be in cluster  $i$  (as  $x_i$  did not change at all). Thanks to strategyproofness, this means cluster  $i$ 's share is still  $y$ .

If  $\sum_{j \neq i} |k(x_j - x'_j)| > 2$ , we make the move from  $x$  to  $x'$  using intermediary steps,  $x^0, \dots, x^s$  such that  $x^j = (x^j_1, \dots, x^j_\ell)$  a share allocation where  $x^j_i = x_i$ ,  $x^0 = x$ ,  $x^s = x'$ , and for  $1 \leq t \leq s$ ,  $\sum_{j \neq i} |k(x^t_j - x^{t-1}_j)| \leq 2$ . Thanks to the argument in the previous paragraph, the allocation for cluster  $i$  stays  $y$  in  $x^1$ , and using the argument again (with  $x^1$  as  $x$  and  $x^2$  as  $x'$ ), it stays the same in  $x^2$ . We apply this argument again and again, until we reach cluster  $i$ 's allocation in  $x^s = x'$  is  $y$  as well.  $\square$

**Proof of Theorem 5:** Suppose there is a rounding of quotas that guarantees selection of  $k$  agents. Let us assume  $k$  clusters and  $k > 3$  is odd. Using Lemma 11, we know that each cluster's slot allocation is fixed according to its share, regardless of other clusters' share. Hence, for each cluster with a share of 1.5, it receives an allocation of either 1 slot or 2.

**Case I: There are 2 clusters which allocate 2 slots when their share is 1.5.** Suppose these 2 clusters have a share of 1.5, another cluster has 0, and any remaining ones 1. Hence we allocated  $k$  shares, but received  $k + 1$  slot allocations.

**Case II: There are 2 clusters which allocate 1 slot when their share is 1.5.** Suppose these 2 clusters have a share of 1.5, another cluster has 0, and any remaining ones 1. Hence we allocated  $k$  shares, but received  $k - 1$  slot allocations.  $\square$

### E. Lemmata for Theorem 7

To prove this theorem, we first need several lemmas. Note that any cluster  $i$  needs to be rounded up with probability of  $s_i - \lfloor s_i \rfloor$ , and rounded down with probability  $\lceil s_i \rceil - s_i$ .

**Lemma 12** *Let  $x = (x_1, \dots, x_\ell) \in \mathbb{N}^\ell$  and let set  $A = \{x' \in \mathbb{N}^\ell \mid \text{in } \alpha \in \mathbb{N} \text{ coordinates } x'_i = x_i + 1. \text{ In the rest } x'_i = x_i\}$ . For any  $\hat{x} = (\hat{x}_1, \dots, \hat{x}_\ell)$  that is a simplex of  $A$  (i.e.,  $\hat{x} = \sum_{a \in A} p_a a$  such that  $\sum_{a \in A} p_a = 1$ ),  $\sum_{i=1}^\ell (\hat{x}_i - x_i) = \alpha$ .*

**Proof:** We shall prove it for any  $\alpha$  by induction over  $\ell$ . The base case is  $\ell = \alpha + 1$ . In that case, any element in set  $A$  has one coordinate where it is equal to  $x$ . Hence, for each coordinate,  $1 - (\hat{x}_i - x_i)$  is the weight assigned to the element of  $A$  where coordinate  $i$  is  $x_i$ . Hence, we know

$$\begin{aligned} \sum_{i=1}^\ell (1 - (\hat{x}_i - x_i)) &= 1 \\ \ell - \sum_{i=1}^\ell (\hat{x}_i - x_i) &= 1 \\ \implies \sum_{i=1}^\ell (\hat{x}_i - x_i) &= \ell - 1 = \alpha \end{aligned}$$

We now assume the lemma is true for  $\ell = h$  ( $h > \alpha$ ), and we shall prove it for  $\ell = h + 1$ . Let  $\hat{x} = \sum_{a \in A} w_a a$ , and let set  $B \subset A$  be all the elements of  $A$  in which coordinate 1 is  $x_1 + 1$ . The weight assigned to these elements is  $\hat{x}_1 - x_1$ , and therefore  $\sum_{a \in (A \setminus B)} w_a = 1 - (\hat{x}_1 - x_1)$ .

We create  $\hat{x}' = \sum_{a \in (A \setminus B)} \frac{1}{1 - (\hat{x}_1 - x_1)} w_a a$ , which is a simplex of elements of  $A$ , and as we can ignore its first coordinate (which remains fixed), and look at it as an element in an  $h$  dimensional space, the induction hypothesis tells us  $\sum_{i=2}^\ell (\hat{x}'_i - x_i) = \alpha$ . Therefore,  $\sum_{i=1}^\ell ((\sum_{a \in (A \setminus B)} w_a a)_i - (\sum_{a \in (A \setminus B)} w_a) x_i) = \alpha(1 - (\hat{x}_1 - x_1))$ .

Every element in  $B$  contributes its weight to  $\alpha$  coordinates (one of them is 1, according to  $B$ 's definition). So for element  $b \in B$ ,  $\sum_{i=1}^\ell ((w_b b + \sum_{a \in (A \setminus B)} w_a a)_i - (w_b + \sum_{a \in (A \setminus B)} w_a) x_i) = \alpha(1 - (\hat{x}_1 - x_1)) + \alpha w_b$ . Adding all the elements of  $B$  (which weigh, together,  $(\hat{x}_1 - x_1)$ ), we get  $\sum_{i=1}^\ell (\hat{x}_i - x_i) = \alpha(1 - (\hat{x}_1 - x_1)) + \alpha(\hat{x}_1 - x_1) = \alpha$ .  $\square$

Note that Lemma 12 is applicable to our algorithm, as we can consider  $x = (\lfloor s_1 \rfloor, \dots, \lfloor s_\ell \rfloor)$  as a basis, and in each allocation the algorithm rounds up  $\alpha$  coordinates, and this rounding up is equivalent to taking  $\alpha$  coordinates, and instead of putting the value from  $x$ , we round up and add 1 to the coordinate.

**Lemma 13** *At no point in the algorithm is a cluster rounded up or down in allocations that, together, have more probability than it should (i.e., for any cluster  $i$ , it is always true that  $p_i \leq s_i - \lfloor s_i \rfloor$  and  $\bar{p} - p_i \leq \lceil s_i \rceil - s_i$ ). Moreover, for any  $i < \text{low}$ , probability that cluster  $i$  is rounded up is  $s_i - \lfloor s_i \rfloor$ . For any  $i > \text{high}$ , probability that cluster  $i$  is rounded down is  $\lceil s_i \rceil - s_i$ .*



**Proof:** Recall the probability cluster  $i$  is rounded up needs to be  $s_i - \lfloor s_i \rfloor$ . When a set of clusters is being rounded up, the probability of the allocation is bounded by  $s_{low} - \lfloor s_{low} \rfloor - p_{low}$  (Line 14), i.e., the probability  $s_{low}$  should be rounded up which still remains to be allocated. Any other cluster  $i$  rounded up in the same allocation has been rounded with  $s_{low}$  in every previous allocation when it has been rounded up (since  $i > low$ ), so  $p_i \leq p_{low}$ . Since the clusters are ordered according to it,  $s_i - \lfloor s_i \rfloor > s_{low} - \lfloor s_{low} \rfloor$ . Hence,  $s_{low} - \lfloor s_{low} \rfloor - p_{low} \leq s_i - \lfloor s_i \rfloor - p_i$ , so no cluster is rounded up more than it should be.

At any point in the algorithm, If  $i < low$ , then there was a stage where  $low = i$ , and line 15 changed  $low$  to  $i + 1$ . However, at that point, cluster  $i$  is rounded up exactly at the additional probability it needed to be rounded up ( $s_i - \lfloor s_i \rfloor - p_i$ ), and no further allocation in the algorithm will round it up.

Similarly, If  $i > high$ , there was a stage where  $high = i$ , and line 18 changed  $high$  to  $i - 1$ . However, at that point, cluster  $i$  is rounded down exactly at the additional probability it needs to be rounded down. It could not have been rounded down too much previously, as every allocation's probability is bounded so that cluster  $high$  will not be rounded down more than it is supposed to ( $\lceil s_{high} \rceil - s_{high}$ ). Once again, once an index is  $high + 1$ , no further allocation will round it down.

For cluster  $i$ ,  $low \leq i \leq high$ , it has only been rounded down when  $s_{high}$  was rounded down as well (though not vice versa) and rounded up when  $s_{low}$  was rounded up (again, not vice versa), and as the  $\lceil s_i \rceil - s_i \geq \lceil s_{high} \rceil - s_{high}$  and  $s_i - \lfloor s_i \rfloor \geq s_{low} - \lfloor s_{low} \rfloor$ , these clusters have not had allocations with more than the probability of them being rounded up or down. Therefore, the amount of probabilities allocated is never larger than 1. Hence, also, for clusters  $i < low$ , as they have received the exact needed probability of being rounded up, and since the sum of allocations does not exceed 1, they have not been rounded down more than needed.  $\square$

## F. Proof of Theorem 7

**Proof:** We first show all the allocations we consider only round up  $\sum_{i=1}^{\ell} (s_i - \lfloor s_i \rfloor)$  clusters, as otherwise, allocations are not allocating exactly  $k$  agents. As long as  $low + \alpha \leq high$  in the algorithm this is trivially true. We now wish to show the situation  $low + \alpha > high$  cannot happen (as that results in too few clusters rounded up).

If  $low + \alpha > high$ , this means there was a stage in which  $low + \alpha = high$ , and then we executed line 15. I.e., cluster  $high$  needs more unallocated probability to be rounded down than cluster  $low$  needs unallocated probability to be rounded up. Moreover, thanks to monotonicity of  $s_i$  vector, we know cluster  $high$  still has need for more allocations with positive probability in which it is rounded up. But that means that if we did advanced  $low$  and assigned all remaining unassigned probability to the allocation rounding up clusters  $low + 1, \dots, \ell$ , we would be rounding them up too much, and for clusters  $low + 1, \dots, high$ , strictly so. But we know (Lemma 13) that for all  $i \leq low$ , we have rounded it up exactly correctly, so looking at the vector  $\hat{x} \in \mathbb{N}^{\ell}$  in which each coordinate is the expected allocation for that cluster, we have:

$$\begin{aligned} \sum_{i=1}^{\ell} (\hat{x}_i - \lfloor s_i \rfloor) &= \sum_{i=1}^{low} (s_i - \lfloor s_i \rfloor) + \sum_{i=low+1}^{\ell} \hat{x}_i - \lfloor s_i \rfloor > \\ &> \sum_{i=1}^{\ell} (s_i - \lfloor s_i \rfloor) = \alpha \end{aligned}$$

This contradicts Lemma 12, as all allocations had exactly  $\alpha$  clusters rounded up. So it cannot be that cluster  $high$  still needed more probability to be rounded down, and therefore, if  $low + \alpha = high$ , line 15 would not have been executed at this point.

Since  $low + \alpha \leq high$  at all times, the algorithm will end when  $low = high$ . Hence, the previous step ended with  $\alpha$  becoming 0 (Line 18) ( $\alpha = 0$  only in this case: otherwise, it means the sum of expected value – that is, probability to be rounded up – over all clusters is above  $\alpha$ : we have too many clusters that need to be rounded up). We now wish to prove that the last step, where the clusters  $high + 1, \dots, \ell$  are rounded up, results in what we desired. Since clusters  $1, \dots, low - 1$  have been allocated the right probability to be rounded up, as well as clusters  $high + 1, \dots, \ell$  (Lemma 13), we only need to verify this for cluster  $low$ . But according to Lemma 12, the probability of  $low$  being rounded up is exactly  $\alpha - \sum_{1 \leq i \leq \ell, i \neq low} (s_i - \lfloor s_i \rfloor)$ , which exactly  $s_{low} - \lfloor s_{low} \rfloor$ , which means cluster  $low$  has the correct allocation.  $\square$

## G. Proof of Theorem 8

**Proof:** For Dollar Raffle and Dollar Partition Raffle, the proof follows a similar path: The mechanism iterates until it chooses  $k$  different agents, which is equivalent to eliminating each selected agent and re-normalizing the dollar partitions, as once some agent is selected we ignore its repeated selection. This re-normalization prevents the mechanism from being strategyproof, as now the probabilities of others matter for each agent. For example, an agent will prefer to contribute to a very strong agent (which, once eliminated, will make our agent's probability increase significantly). Suppose  $k = 2$ , if Dollar Raffle (Dollar Partition Raffle), suppose all agents (clusters) except  $b_1, b_2, b_3$  allocate their points equally between those 3.  $b_1$  divides its point equally between  $b_2$  and  $b_3$ , as does  $b_2$  between  $b_1$  and  $b_3$ . Suppose  $b_3$  believes it should also divide its point equally between  $b_1$  and  $b_2$ . In that case, it has a probability  $\frac{1}{3}$  of being selected first, and a probability of  $\frac{1}{3}$  of being selected second, ultimately,  $\frac{2}{3}$ . But if agent (in)  $b_3$  decides to give its point fully to cluster  $b_1$ , the probability of  $b_3$  being selected first does not change. But the probability of  $b_1$  being selected and then  $b_3$  are  $(\frac{1}{3} + \frac{1}{2n}) \frac{\frac{1}{3}}{\frac{2}{3} - \frac{1}{2n}}$ , and the probability of  $b_2$  being selected and then  $b_1$  are  $(\frac{1}{3} - \frac{1}{2n}) \frac{\frac{1}{3}}{\frac{2}{3} + \frac{1}{2n}}$ . The sum of these is more than  $\frac{1}{3}$ , hence doing so would improve agent (in cluster)  $b_3$  chances of being selected. This proof can easily be extended to any additional  $k$ .

The proof of this theorem carries on to the various mechanisms presented for  $k = 1$  (e.g., [6]): simply running the algorithm several times destroys its strategyproofness. This is true even for mechanisms that are strategyproof for  $k = 1$ , as long as any agent has the power to influence the outcome (i.e., not purely random, a dictatorship, or a combination of both).

For Top Dollar, agents are  $a_1, \dots, a_{k+1}$ . Agents  $a_1, \dots, a_{k-1}$  allocate each of their points by giving  $\frac{1}{k} - \frac{1}{k^2}$  to agent  $a_{k+1}$ ,  $\frac{1}{k^2}$  to agent  $a_k$  and  $\frac{1}{k}$  to all other agents. Agent  $a_k$  gives  $\frac{1}{k}$  to all other agents. Agent  $a_{k+1}$  would like to allocate its point to agent  $a_k$ , but that would mean it would not be selected itself. Giving its point to other agents will mean it will be, contradicting strategyproofness.  $\square$